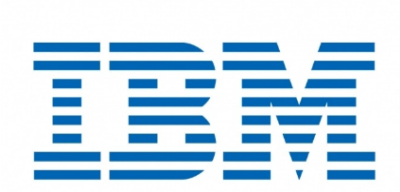

Data Insight Selfmon - ps-di-selfmon

Overview

DataInsight is part of the SevOne NPM architecture. However, there is no OOTB monitoring of DI. This add on provides a solution to monitor DI metrics and ingest them into the SevOne appliance.

Solution

This project is an add-on for selfmon ofor Data Insight. The planned solution will query prometheus on DataInsight for the configured metrics, and the data on prometheus will be ingested into SevOne.

The main functionality of this add-on is,

- To connect to the DataInsight configured.
- To query prometheus for each of the metrics mentioned in the configuration file
- Convert the queried data into Device- Object - Indicator format
- Ingest the above data into SevOne cluster

Table of Contents

Prerequisites

Before you begin, ensure you have met the following requirements:

1. SevOne V3 APIs credentials [Tested for SevOne 6.7 and 6.8].
2. DataInsight Prometheus credentials [This is usually datainsight / datainsight]
3. Metrics that needs to be queried on Prometheus.

Installation

To install this project, follow these steps:

1. Create a directory ~/ps-di-selfmon on the cluster master of the SevOne cluster.
2. Download the RPM IBM-SevOne-NPM-PS-Plugin-ps-di-selfmon--1.el8.x86_64.rpm to the ~/ps-di-selfmon directory
3. Install the RPM

```
yum localinstall IBM-SevOne-NPM-PS-Plugin-ps-di-selfmon-
<version>-1.el8.x86_64.rpm
```

4. Update `/opt/sevone-uc/ps-di-selfmon/config.json` to include your SevOne API credentials, DI Prometheus credentials other necessary configuration details. You can use 'UseSShKeys = 1' to use sshKeys to login. If not, you can set 'sshPassword'

```
 {
     "ApplianceDetails": [
     {
         "IPAddress": "<ClusterMaster IP Address>",
         "UserName": "<API username",
         "Password": "<API password>",
         "sshUserName": "<ssh userName>", # Required
         "sshPassword": "<ssh Password>", # To be set only if
UseSSHKeys = 0
         "UseSShKeys": 1, # Can take value 0 or 1
         "Type": "NMS"
     }
     ]
 }
```

```
    "DIDetails": [
    {
        "IPAddress": "<IPAddress>",
        "UserName": "datainsight",
        "Password": "datainsight",
        "Type": "Prometheus",
```

```
       "Metrics": {
           "<ObjectType1>": [
               {
                   "query": "<prom Query1>",
                   "indicatorName": "" # Optional
                   "type": "GAUGE/COUNTER64",
                   "units": "Number/Seconds/Bytes/.."
               },
               {
                   "query": "<prom Query2>",
                   "type": "GAUGE/COUNTER64",
                   "units": "Number/Seconds/Bytes/.."
               },
               {
                   "query": "<prom Query3>",
                   "type": "GAUGE/COUNTER64",
                   "units": "Number/Seconds/Bytes/.."
```

```
            }
        ],
        "<ObjectType2>": [
            {
                "query": "<prom Query4>",
                "type": "GAUGE/COUNTER64",
                "units": "Number/Seconds/Bytes/.."
            },
            {
                "query": "<prom Query5>",
                "type": "GAUGE/COUNTER64",
                "units": "Number/Seconds/Bytes/.."
            }
        ]
```

```
"interval": 300, # To set the interval this add-on has to run
"LogLevel": "INFO",
"MaxLogFileSize": "10485760",
```

5. Run the following command to install the add-on on the Cluster master and all the peers on the cluster Install:

```
/opt/sevone-uc/ps-di-selfmon/bin/run-di-selfmon.sh
```

Maintanance

1. To run the add-on, execute the following command:

```
/opt/sevone-uc/ps-di-selfmon/bin/run-di-selfmon.sh
```

   This will run the add-on

2. To stop the add-on, execute the following command:

```
docker-compose -f /opt/sevone-uc/ps-di-selfmon/docker-compose.yml down
```

   This will stop the add-on

Verification

To verify if the application is running fine

1. Check if the docker container is running

```
docker ps -f name=ps-di-selfmon
```

2. Check the log files at

```
/var/log/SevOne/ps-di-selfmon/ps-di-selfmon.log
```

Important Metrics that can be monitored

1. Pod Status

```
pod_status: promquery - kube_pod_status_phase{phase=\"Running\"}
pod_status = 0 : Pod is not running
pod_status = 1 : Pod is running
```

2. Node / WorkerNode Status

```
node_status: promquery -
kube_node_status_condition{condition=\"Ready\", status=\"true\"}
node_status = 0 : Node is not Ready
node_status = 1 : Node is  Ready
```

3. Disk, Memory Pressure on DI nodes / WorkerNodes

```
disk_pressure: promquery -
kube_node_status_condition{condition=\"DiskPressure\",
status=\"true\"}

memory_pressure: promquery -
kube_node_status_condition{condition=\"MemoryPressure\",
status=\"true\"}

disk_pressure / memory_pressure  = 0 : Disk is OK / Memory is OK
disk_pressure / memory_pressure  = 1 : Disk is not OK / Memory is not
OK
```

4. Deployment status

```
Replicas Unavailable: promquery : kube_deployment_spec_replicas -
kube_deployment_status_replicas_available
Replicas Unavailable = 0 : Deployment is OK. All Replicas are
available
Replicas Unavailable != 0 : Deployment is not OK. Some replicas are
unavailable. If the replicas are restarting, then this metric might
become non-zero temporarily.
```

5. Certificate status

```
Time to Expire in weeks: promquery : kube_deployment_spec_replicas -
kube_deployment_status_replicas_available
Time to Expire in weeks > 46 : Certificate expire time is more than a
year
Time to Expire in weeks < 46 : Certificate will expire within a year.
```